Lyapunov-Derived Control and Adaptive Update Laws for Inner and Outer Layer Weights of a Deep Neural Network

Omkar Sudhir Patil, *Graduate Student Member, IEEE*, Duc M. Le, Max L. Greene, and Warren E. Dixon, *Fellow, IEEE*

***Abstract*—Lyapunov-based real-time update laws are well-known for neural network (NN)-based adaptive controllers that control nonlinear dynamical systems using single-hidden-layer NNs. However, developing real-time weight update laws for deep NNs (DNNs) remains an open question. This letter presents the first result with Lyapunov-based real-time weight adaptation laws for each layer of a feedforward DNN-based control architecture, with stability guarantees. Additionally, the developed method allows nonsmooth activation functions to be used in the DNN to facilitate improved transient performance. A nonsmooth Lyapunov-based stability analysis proves global asymptotic tracking error convergence. Simulation results are provided for a nonlinear system using DNNs with leaky rectified linear unit (LReLU) and hyperbolic tangent activation functions to demonstrate the efficacy of the developed method.**

***Index Terms*—Deep neural networks, deep learning, adaptive control, Lyapunov methods, nonlinear control systems.**

I. Introduction

NEURAL networks (NNs) are universal function approximators that are capable of modeling continuous functions over a compact domain [1]. Although NNs with a single hidden-layer are capable of approximating general nonlinear functions, deep NNs (DNNs) provide improved performance [2]. Moreover, DNNs are exponentially more expressive than shallow NNs in terms of the total number of neurons required to achieve the same accuracy in function approximation [3].

Manuscript received September 14, 2021; revised November 24, 2021; accepted December 9, 2021. Date of publication December 14, 2021; date of current version December 22, 2021. This work was supported in part by NSF under Award 1762829; in part by the Office of Naval Research under Grant N00014-13-1-0151; and in part by the Air Force Office of Scientific Research (AFOSR) under Award FA9550-18-1-0109 and Award FA9550-19-1-0169. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Omkar Sudhir Patil.)*

The authors are with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: patilomkarsudhir@ufl.edu; ledan50@ufl.edu; maxgreene12@ufl.edu; wdixon@ufl.edu).

Digital Object Identifier 10.1109/LCSYS.2021.3134914

Motivated by recent advances in DNNs, researchers have explored the use of DNN-based control architectures. DNN-based techniques often employ optimization methods to train the DNN weights by minimizing a loss function over a training dataset [4]. Results in [5]–[7] utilize such offline DNN training techniques to approximate explicit model predictive control laws. However, such offline methods pose limitations since training typically requires large amounts of data, and the resulting feedforward terms are implemented as an open-loop approximator based on the offline training.

In contrast to offline training, NN weight update laws derived from Lyapunov-based stability analysis methods have been developed to adjust the NN weights in real-time as an adaptive closed-loop feedforward term [8]. Although NN-based adaptive architectures are well-established, these methods only apply to NNs with a single hidden-layer. The complex nature of DNNs being nested nonlinear parameterizations of inner-layer activation functions, weights, and bias terms presents challenges that preclude development of real-time adaptation laws with Lyapunov-based methods.

Motivated by function approximation abilities of DNNs, emerging results in [9]–[12] develop real-time DNN-based adaptive architectures. In [9] and [10], real-time DNN-based adaptive architectures are developed for model reference adaptive control. Similarly, results in [11] generalize the DNN-based adaptive architecture to general nonlinear systems. However, such results only update the output-layer weights in real-time. While the output-layer weights are updated in real-time, data is collected and used to train the inner-layer weights iteratively over discrete training periods via traditional offline techniques. In [12], insights are provided into the development of real-time adaptive weight update laws for individual layers of a feedforward DNN based on a modular design. Modular adaptive designs develop mild constraints on the adaptation laws and provide stability guarantees based on the worst-case scenario of the developed constraints. Although the modular adaptive approach provides constraints on the weight update laws, these constraints are only sufficient and lack insights on how to best design the inner-layer weight adaptation laws.

This letter presents the first result with Lyapunov-based real-time weight adaptation laws for each layer of a DNN. A

general uncertain nonlinear system is considered. To address the challenges posed by nested nonlinear parameterizations of the inner-layer DNN weights, we develop a recursive representation of the inner-layer DNN structure to facilitate the analysis. Then, a Taylor's first order approximation of the uncertainty is recursively derived. Subsequently, the update laws are derived from a Lyapunov-based stability analysis, in which the first-order terms are canceled by the weight update law-based terms. The remaining terms in the Lyapunov-based analysis are eliminated using a robust control approach.

The adaptation laws developed in this letter depend on gradients of activation functions. The adaptation laws contain discontinuities if an activation function with a discontinuous gradient is used in the DNN architecture. Nonsmooth activation functions such as rectified linear units (ReLUs), leaky ReLUs (LReLUs) [13], maxout [14], etc. are often preferred over sigmoidal activation functions, since they empirically exhibit improved function approximation performance while also overcoming the vanishing gradient problem [4, Ch. 6]. As previously noted, the discontinuities in gradients of these activation functions pose difficulties in facilitating the standard Lyapunov-based analysis. In this letter, a nonsmooth analysis is performed to address the challenges of including nonsmooth activation functions. The nonsmooth Lyapunov-based analysis guarantees global asymptotic tracking error convergence. Simulation results are provided for a nonlinear system using DNNs with leaky ReLU and hyperbolic tangent activation functions to demonstrate the efficacy of the developed method. A comparison of a DNN with leaky ReLU activation functions to a DNN with hyperbolic tangent activation functions shows improved tracking and function approximation performance while using the DNN with leaky ReLU activation functions.

Notation and Preliminaries: The space of essentially bounded Lebesgue measurable functions is denoted by $\mathcal{L}_\infty$. The right-to-left matrix product operator is represented by $\prod^{\curvearrowleft m}$, i.e., $\prod_{p=1}^{\curvearrowleft m} A_p = A_m \ldots A_2 A_1$ and $\prod_{p=a}^{\curvearrowleft m} A_p = 1$ if $a > m$. The vectorization operator is denoted by $\text{vec}(\cdot)$, i.e., given $A \triangleq [a_{i,j}] \in \mathbb{R}^{n\times m}$, $\text{vec}(A) \triangleq [a_{1,1}, \ldots, a_{1,m}, \ldots, a_{n,1}, \ldots, a_{n,m}]^T$. The p-norm is denoted by $\|\cdot\|_p$, where the subscript is suppressed when $p = 2$. The Frobenius norm is denoted by $\|\cdot\|_F \triangleq \|\text{vec}(\cdot)\|$. The Kronecker product is denoted by $\otimes$. Function compositions are denoted using the symbol $\circ$, e.g., $(g \circ h)(x) = g(h(x))$, given suitable functions f and g. The Filippov set-valued map defined in [15, Equation 2b] is denoted by $K[\cdot]$. Consider a Lebesgue measurable and locally essentially bounded function $h : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. Then, the function $y : \mathcal{I} \to \mathbb{R}^n$ is called a *Filippov solution* of $\dot{y} = h(y, t)$ on the interval $\mathcal{I} \subseteq \mathbb{R}_{\geq 0}$ if y is absolutely continuous on $\mathcal{I}$ and $\dot{y} \in K[h](y, t)$ for almost all $t \in \mathcal{I}$. The notation $F : A \rightrightarrows B$ denotes a set-valued map from set A to set B. Given some functions f and g, the notation $f(y) = \mathcal{O}^m(g(y))$ means that there exists some constants $M \in \mathbb{R}_{>0}$ and $y_0 \in \mathbb{R}$ such that $\|f(y)\| \leq M\|g(y)\|^m$ for all $y \geq y_0$.

Fact 1 [16, Proposition 7.1.9]: Given any $A \in \mathbb{R}^{p\times a}$, $B \in \mathbb{R}^{a\times r}$, and $C \in \mathbb{R}^{r\times s}$, $\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B)$.

II. Unknown System Dynamics and Control Design

Consider a control-affine nonlinear dynamic system modeled as

$$\dot{x} = f(x) + u, \tag{1}$$

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes a Filippov solution[1] to (1), $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown differentiable function, and $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ denotes a control input. The control objective is to track a user-defined reference trajectory $x_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. The reference trajectory is designed to be continuously differentiable, such that $x_d(t) \in \Omega$, $\forall t \in \mathbb{R}_{\geq 0}$, and $\dot{x}_d \in \mathcal{L}_\infty$, where $\Omega \subset \mathbb{R}^n$ denotes a known compact set. To quantify the tracking objective, the tracking error $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is defined as

$$e \triangleq x - x_d. \tag{2}$$

A. Deep Neural Network Architecture

A variety of DNN architectures are known to approximate any given continuous function on a compact set, based on universal approximation theorems that can be invoked case-by-case for DNN architectures [18]. Let $\Phi : \mathbb{R}^n \times \mathbb{R}^{L_0\times L_1} \times \ldots \times \mathbb{R}^{L_k\times L_{k+1}} \to \mathbb{R}^n$ denote the feedforward DNN architecture defined as

$$\Phi(x_d, V_0, V_1, \ldots, V_k) \triangleq \left(V_k^T \phi_k \circ \cdots \circ V_1^T \phi_1\right)\left(V_0^T x_{da}\right), \tag{3}$$

where $x_{da} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n+1}$ denotes the augmented desired state $x_{da} \triangleq [x_d^T \quad 1]^T$, and $k \in \mathbb{N}$ denotes the total number of hidden-layers. The matrix of weights and biases at the j^{th} layer is denoted by $V_j \in \mathbb{R}^{L_j\times L_{j+1}}$, where $L_j \in \mathbb{N}$ denotes the number of nodes in the j^{th} inner-layer for all $j \in \{0, \ldots, k\}$, with $L_0 \triangleq n + 1$ and $L_{k+1} \triangleq n$. The vector of smooth[2] activation functions at the j^{th} layer is denoted by $\phi_j : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j}$. If the DNN involves multiple types of activation functions at each layer, then ϕ_j may be represented as $\phi_j \triangleq [\varsigma_{j,1} \cdots \varsigma_{j,L_j-1} \quad 1]^T$, where $\varsigma_{j,i} : \mathbb{R} \to \mathbb{R}$ denotes the activation function at the i^{th} node of j^{th} layer. Note that x_{da} and ϕ_j are augmented with 1 to facilitate the inclusion of a bias term. The DNN architecture in (3) can also be represented recursively as

$$\Phi_j \triangleq \begin{cases} V_j^T \phi_j\left(\Phi_{j-1}\right), & j \in \{1, \ldots, k\}, \\ V_0^T x_{da}, & j = 0, \end{cases} \tag{4}$$

and $\Phi(x_d, V_0, \ldots, V_k) = \Phi_k$, where $\Phi_j : \mathbb{R}^n \times \mathbb{R}^{L_0\times L_1} \times \ldots \times \mathbb{R}^{L_j\times L_{j+1}} \to \mathbb{R}^{L_{j+1}}$ denotes $(x_d, V_0, \ldots, V_j) \mapsto \Phi_j(x_d, V_0, \ldots, V_j)$. The universal function approximation property states that the function space of DNNs given by (3) is dense in $\mathcal{C}(\Omega)$ [18, Th. 3.2], where $\mathcal{C}(\Omega)$ denotes the space of functions continuous over Ω. For any given

[1] We consider generalized solutions such as Filippov or Krasovskii solutions instead of classical solutions to facilitate a nonsmooth control design. These solutions are guaranteed to exist for nonsmooth systems with Lebesgue measurable and locally essentially bounded right-hand-sides [17, Proposition 3], whereas classical solutions might not exist.

[2] Although ϕ_j is defined as a smooth function, the subsequent analysis allows the inclusion of nonsmooth activation functions by modeling them via a switching mechanism involving smooth functions.

$f \in \mathcal{C}(\Omega)$ and prescribed $\bar{\varepsilon} \in \mathbb{R}_{>0}$, there exist some $k, L_j \in \mathbb{N}$, and corresponding ideal weights and biases, $V_j^* \in \mathbb{R}^{L_j \times L_{j+1}}$, $\forall j \in \{0, \ldots, k\}$, such that $\sup_{x_d \in \Omega} \|f(x_d) - \Phi(x_d, V_0^*, V_1^*, \ldots, V_k^*)\| \leq \bar{\varepsilon}$. Then the unknown function in (1) can be modeled as

$$f(x_d) = \Phi(x_d, V_0^*, V_1^*, \ldots, V_k^*) + \varepsilon(x_d), \tag{5}$$

where $\varepsilon : \mathbb{R}^n \to \mathbb{R}^n$ denotes the unknown function approximation error such that $\sup_{x_d \in \Omega} \|\varepsilon(x_d)\| \leq \bar{\varepsilon}$. It is assumed there exists a known constant $\overline{V} \in \mathbb{R}_{>0}$ such that $\sup_{x_d \in \Omega, \forall j} \|V_j^*\|_F \leq \overline{V}$ (cf., [19, Assumption 1]).

B. Control Law Development

The universal approximation property makes DNN-based adaptive control architectures well-suited for unknown dynamics, as in (1) where $f(\cdot)$ is unknown [18, Th. 3.2]. The adaptive feedforward DNN term is designed as $\hat{\Phi} \triangleq \Phi(x_d, \hat{V}_0, \ldots, \hat{V}_k)$, where $\hat{V}_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_j \times L_{j+1}}$ for all $j \in \{0, \ldots, k\}$ denotes the estimated weight matrix for the j^{th} layer. The weight estimation error of the ideal inner-layer weights $\widetilde{V}_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_j \times L_{j+1}}$ for all $j \in \{0, \ldots, k\}$ is defined as $\widetilde{V}_j \triangleq V_j^* - \hat{V}_j$. The gradient of the activation function vector at the j^{th} layer is denoted as $\phi_j' : \mathbb{R}^{L_j} \to \mathbb{R}^{L_j \times L_j}$, and $\phi_j'(y) \triangleq \frac{\partial}{\partial z}\phi_j(z)|_{z=y}$, $\forall y \in \mathbb{R}^{L_j}$. To facilitate the subsequent stability analysis, let the function $f_e : \mathbb{R}^n \times \Omega \to \mathbb{R}^n$ be defined as $f_e \triangleq f(x) - f(x_d)$. By [20, Lemma 5], the function $(x, x_d) \mapsto f_e$ is bounded as $\|f_e\| \leq \rho(\|e\|)\|e\|$ for all $x \in \mathbb{R}^n$ and $x_d \in \Omega$, where $\rho : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a known strictly increasing function. Based on the subsequent stability analysis, the control input is designed as

$$u \triangleq \dot{x}_d - \rho(\|e\|)e - k_1 e - k_s \text{sgn}(e) - \hat{\Phi}, \tag{6}$$

where $k_1, k_s \in \mathbb{R}_{>0}$ are user-defined control gains, and $\text{sgn}(\cdot)$ denotes the vector signum function. The following short-hand notations are introduced for brevity in the subsequent analysis: $\Phi_j^* \triangleq \Phi_j(x_d, V_0^*, \ldots, V_j^*)$, $\hat{\Phi}_j \triangleq \Phi_j(x_d, \hat{V}_0, \ldots, \hat{V}_j)$, $\widetilde{\Phi}_j \triangleq \Phi_j^* - \hat{\Phi}_j$, $\Phi^* \triangleq \Phi_k^*$, $\widetilde{\Phi} \triangleq \widetilde{\Phi}_k = \Phi^* - \hat{\Phi}$, $\phi_j^* \triangleq \phi_j(\Phi_{j-1}^*)$, $\hat{\phi}_j \triangleq \phi_j(\hat{\Phi}_{j-1})$, and $\hat{\phi}_j' \triangleq \phi_j'(\hat{\Phi}_{j-1})$.

Based on the subsequent analysis, the input layer weight adaptation law is designed as

$$\text{vec}(\dot{\hat{V}}_0) \triangleq \text{proj}(\Gamma_0((\overset{\curvearrowleft}{\prod_{l=1}^{k}} \hat{V}_l^T \hat{\phi}_l')(I_{L_1} \otimes x_{da}^T))^T e), \tag{7}$$

and the j^{th} layer weight adaptation law is designed as

$$\text{vec}(\dot{\hat{V}}_j) \triangleq \text{proj}(\Gamma_j((\overset{\curvearrowleft}{\prod_{l=j+1}^{k}} \hat{V}_l^T \hat{\phi}_l')(I_{L_{j+1}} \otimes \hat{\phi}_j^T))^T e), \tag{8}$$

$\forall j \in \{1, \ldots, k\}$, where $\Gamma_j \in \mathbb{R}^{L_j L_{j+1} \times L_j L_{j+1}}$ is a positive-definite adaptation gain matrix for all $j \in \{0, \ldots, k\}$. The operator $\text{proj}(\cdot)$ denotes the projection operator defined in [21, Appendix E, eq. (E.4)], which is used to ensure $\hat{V}_j(t) \in \mathcal{B}_j \triangleq \{\theta \in \mathbb{R}^{L_j L_{j+1}} : \|\theta\|_F \leq \overline{V}\}$, $\forall (t, j) \in \mathbb{R}_{\geq 0} \times \{0, 1, \ldots, k\}$.

III. Stability Analysis

A. Closed-Loop Error System Development

Subtracting $\hat{\Phi}_j$ from Φ_j^*, using (4), adding and subtracting $V_j^{*T}\hat{\phi}_j$, and rearranging terms yields

$$\widetilde{\Phi}_j = \widetilde{V}_j^T \hat{\phi}_j + V_j^{*T}(\phi_j^* - \hat{\phi}_j), \tag{9}$$

$\forall j \in \{1, \ldots, k\}$, and $\widetilde{\Phi}_0 = \widetilde{V}_0^T x_{da}$. Using (1), (2), and (6) yields the closed-loop error system

$$\dot{e} = f_e + \widetilde{\Phi} + \varepsilon(x_d) - \rho(\|e\|)e - k_1 e - k_s \text{sgn}(e). \tag{10}$$

The term $\widetilde{\Phi}$ in (10) has a nested nonlinear parameterization in V_j^* and $\hat{V}_j$, which precludes the application of traditional analysis techniques that are used for linearly parameterized adaptive systems. A first-order Taylor series approximation is developed in [19] to overcome the challenges presented by the nonlinear parameterization for three-layer neural networks. To overcome the nested structure of nonlinear parameterization in DNNs, we use a recursive approach to develop a first-order Taylor series approximation for ϕ_j^*, $\widetilde{\Phi}_j$, and $\widetilde{\Phi}$. Using the first-order Taylor series approximation in [19, eq. (22)] yields

$$\phi_j^* = \hat{\phi}_j + \hat{\phi}_j' \widetilde{\Phi}_{j-1} + \mathcal{O}^2(\widetilde{\Phi}_{j-1}), \tag{11}$$

$\forall j \in \{1, \ldots, k\}$. Substituting (11) into (9), adding and subtracting $\hat{V}_j^T \hat{\phi}_j' \widetilde{\Phi}_{j-1}$, and rearranging terms yields

$$\widetilde{\Phi}_j = \widetilde{V}_j^T \hat{\phi}_j + \hat{V}_j^T \hat{\phi}_j' \widetilde{\Phi}_{j-1} + \Delta_j, \tag{12}$$

where $\Delta_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_{j+1}}$ is defined as

$$\Delta_j \triangleq \widetilde{V}_j^T \hat{\phi}_j' \widetilde{\Phi}_{j-1} + V_j^{*T} \mathcal{O}^2(\widetilde{\Phi}_{j-1}), \tag{13}$$

$\forall j \in \{1, \ldots, k\}$. Since the term $\widetilde{V}_j^T \hat{\phi}_j$ is a vector, $\widetilde{V}_j^T \hat{\phi}_j = \text{vec}(\widetilde{V}_j^T \hat{\phi}_j) = \text{vec}(\hat{\phi}_j^T \widetilde{V}_j) = \text{vec}(\hat{\phi}_j^T \widetilde{V}_j I_{L_{j+1}})$. Applying Fact 1 on $\text{vec}(\hat{\phi}_j^T \widetilde{V}_j I_{L_{j+1}})$ yields

$$\widetilde{V}_j^T \hat{\phi}_j = \left(I_{L_{j+1}} \otimes \hat{\phi}_j^T\right)\text{vec}(\widetilde{V}_j). \tag{14}$$

Substituting (14) into (12) yields the recursive representation

$$\widetilde{\Phi}_j = \left(I_{L_{j+1}} \otimes \hat{\phi}_j^T\right)\text{vec}(\widetilde{V}_j) + \hat{V}_j^T \hat{\phi}_j' \widetilde{\Phi}_{j-1} + \Delta_j, \tag{15}$$

$\forall j \in \{1, \ldots, k\}$. To facilitate the subsequent analysis, the following lemma yields a generalized expression for $\widetilde{\Phi}_j$.

Lemma 1: For all $j \in \{0, \ldots, k\}$, the term $\widetilde{\Phi}_j$ can be expressed as

$$\begin{aligned}\widetilde{\Phi}_j = &\sum_{i=1}^{j}\left(\overset{\curvearrowleft}{\prod_{l=i+1}^{j}} \hat{V}_l^T \hat{\phi}_l'\right)\left(I_{L_{i+1}} \otimes \hat{\phi}_i^T\right)\text{vec}(\widetilde{V}_i) \\ &+ \left(\overset{\curvearrowleft}{\prod_{l=1}^{j}} \hat{V}_l^T \hat{\phi}_l'\right)(I_{L_1} \otimes x_{da}^T)\text{vec}(\widetilde{V}_0) \\ &+ \sum_{i=1}^{j}\left(\overset{\curvearrowleft}{\prod_{l=i+1}^{j}} \hat{V}_l^T \hat{\phi}_l'\right)\Delta_i.\end{aligned} \tag{16}$$

Proof: See the Appendix. ∎

B. Nonsmooth Analysis

Let $\Xi_0 \triangleq \overset{\curvearrowleft k}{\prod_{l=1}} \hat{V}_l^T \hat{\phi}_l'$, $\Xi_0 \triangleq \overset{\curvearrowleft k}{\prod_{l=j+1}} \hat{V}_l^T \hat{\phi}_l'$, $\Lambda_0 \triangleq \Xi_0 (I_{L_1} \otimes x_{da}^T)$, and $\Lambda_j \triangleq \Xi_j (I_{L_{j+1}} \otimes \hat{\phi}_j^T)$, $\forall j \in \{1, \ldots, k\}$, for notational brevity, where $\Xi_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times L_{j+1}}$ and $\Lambda_j : \mathbb{R}_{\geq 0} \to \mathbb{R}^{L_{j+1} \times L_j L_{j+1}}$, respectively, $\forall j \in \{0, \ldots, k\}$. The subsequent analysis is structured to account for nonsmooth systems. Thus, state-dependent switching between smooth activation functions can also be considered in the analysis. Specifically, a nonsmooth activation function with a finite number of discontinuities in its gradient can be modeled by a switched function involving a collection of smooth activation functions. Let $\sigma \in \mathcal{N}$ denote the switching index considering the total number of switching between activation functions in the entire DNN, where $\mathcal{N} \subset \mathbb{N}$ denotes the set of all possible switching indices. Then, the function approximation in (5) can be represented as $f(x_d) = \Phi_{k,\sigma}(x_d, V_0^*, V_1^*, \ldots, V_k^*) + \varepsilon_\sigma(x_d)$, such that $(x_d, V_0, \ldots, V_j) \mapsto \Phi_{k,\sigma}(x_d, V_0, \ldots, V_j)$ is smooth for each σ with the corresponding approximation error $\varepsilon_\sigma(x_d)$. Thus, ϕ_j^*, $\hat{\phi}_j$, $\hat{\phi}_j'$, $\widetilde{\Phi}_j$ Ξ_j, Λ_j, and Δ_j can also be represented as the switched functions $\phi_{j,\sigma}^*$, $\hat{\phi}_{j,\sigma}$, $\hat{\phi}_{j,\sigma}'$, $\widetilde{\Phi}_{j,\sigma}$, $\Xi_{j,\sigma}$, $\Lambda_{j,\sigma}$, and $\Delta_{j,\sigma}$, respectively, such that they are continuous for each σ. It is assumed that the bound $\sup_{x_d \in \Omega} \|\varepsilon_\sigma(x_d)\| \leq \overline{\varepsilon}$ holds for all $\sigma \in \mathcal{N}$. Using Lemma 1 yields $\widetilde{\Phi} = \widetilde{\Phi}_{k,\sigma} = \sum_{j=0}^k \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j) + \sum_{j=1}^k \Xi_{j,\sigma} \Delta_{j,\sigma}$. Substituting $\widetilde{\Phi}$ into (10) yields

$$\dot{e} = f_e + \sum_{j=0}^k \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j) + \sum_{j=1}^k \Xi_{j,\sigma} \Delta_{j,\sigma} + \varepsilon_\sigma(x_d) - \rho(\|e\|)e - k_1 e - k_s \mathrm{sgn}(e). \tag{17}$$

Additionally, the adaptation laws in (7) and (8) can be represented using $\mathrm{vec}(\dot{\hat{V}}_j) \triangleq \mathrm{proj}(\Gamma_j \Lambda_{j,\sigma}^T e)$, $\forall j \in \{0, \ldots, k\}$. Consequently, $\mathrm{vec}(\dot{\widetilde{V}}_j) = -\mathrm{proj}(\Gamma_j \Lambda_{j,\sigma}^T e)$, $\forall j \in \{0, \ldots, k\}$. Since $\|V_j^*\|_F \leq \overline{V}$ and $\|\hat{V}_j\|_F \leq \overline{V}$, $\forall j \in \{0, \ldots, k\}$, it follows that $\|\widetilde{V}_j\|_F = \|V_j^* - \hat{V}_j\|_F \leq 2\overline{V}$. Moreover, since x_{da} is bounded, and $\phi_{j,\sigma}$ and $\phi_{j,\sigma}'$ are continuous for each $\sigma \in \mathcal{N}$, it follows from (9) that $\phi_{j,\sigma}^*$, $\hat{\phi}_{j,\sigma}$, $\hat{\phi}_{j,\sigma}'$, $\widetilde{\Phi}_{j,\sigma}$, and $\Xi_{j,\sigma}$ can be bounded by known constants for all $(j, \sigma) \in \{0, \ldots, k\} \times \mathcal{N}$. Therefore, based on (13), Δ_j can be bounded by known constants for all $j \in \{1, \ldots, k\}$, and it follows that there exists a known constant $c \in \mathbb{R}_{>0}$ such that

$$\left\| \sum_{i=1}^k \Xi_{i,\sigma} \Delta_{i,\sigma} \right\| \leq c. \tag{18}$$

Let $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^\Psi$ denote the concatenated function, $z \triangleq [e^T, \mathrm{vec}(\widetilde{V}_0)^T, \ldots, \mathrm{vec}(\widetilde{V}_k)^T]^T$, where $\Psi \triangleq n + \sum_{j=0}^k L_j L_{j+1}$ is defined for notational brevity. Let $w_\sigma : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \to \mathbb{R}^\Psi$ denote the concatenated right hand sides of (17) and $\mathrm{vec}(\dot{\widetilde{V}}_j) = -\mathrm{proj}(\Gamma_j \Lambda_{j,\sigma}^T e)$. Then (17) and $\mathrm{vec}(\dot{\widetilde{V}}_j)$ can be represented by the collection of subsystems $\dot{z} = w_\sigma(z, t)$, and the corresponding switched system is represented by

$$\dot{z} = w_{\varrho(z,t)}(z, t), \tag{19}$$

where $\varrho : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \to \mathcal{N}$ denotes a state-dependent switching signal that satisfies [22, Assumption 1].[3] Based on the result in [22], we establish the invariance properties of (19) by establishing the invariance properties of $\dot{z} = w_\sigma(z, t)$ for each $\sigma \in \mathcal{N}$. Let $F_\sigma : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \rightrightarrows \mathbb{R}^\Psi$ denote $K[w_\sigma](z, t)$. Then $F_\sigma(z, t) \subseteq F_\sigma'(z, t)$, where $F_\sigma' : \mathbb{R}^\Psi \times \mathbb{R}_{\geq 0} \rightrightarrows \mathbb{R}^\Psi$ is defined as $F_\sigma'(z, t) \triangleq [\{\sum_{j=0}^k \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j) + \sum_{j=1}^k \Xi_{j,\sigma} \Delta_{j,\sigma} + f_e + \varepsilon_\sigma(x_d) - \rho(\|e\|)e - k_1 e\} - k_s K[\mathrm{sgn}](e); -K[\mathrm{proj}](\Gamma_0 \Lambda_{0,\sigma}^T e); \ldots; -K[\mathrm{proj}](\Gamma_k \Lambda_{k,\sigma}^T e)]$.

Theorem 1: For the dynamical system in (1), the controller in (6) and the adaptation laws in (7) and (8) ensure global asymptotic tracking error convergence in the sense that $\lim_{t \to \infty} \|e(t)\| = 0$, $\forall (e(0), \hat{V}_0, \ldots, \hat{V}_k) \in \mathbb{R}^n \times \mathcal{B}_0 \times \ldots \times \mathcal{B}_k$, provided the gain condition $k_s > \overline{\varepsilon} + c$ is satisfied.

Proof: Consider the candidate common Lyapunov function $\mathcal{V}_L : \mathbb{R}^\Psi \to \mathbb{R}_{\geq 0}$ defined as

$$\mathcal{V}_L(z) \triangleq \frac{1}{2} e^T e + \frac{1}{2} \sum_{j=0}^k \mathrm{vec}(\widetilde{V}_j)^T \Gamma_j^{-1} \mathrm{vec}(\widetilde{V}_j), \tag{20}$$

which satisfies the inequality $\underline{\alpha} \|z\|^2 \leq \mathcal{V}_L(z) \leq \overline{\alpha} \|z\|^2$, where $\underline{\alpha}, \overline{\alpha} \in \mathbb{R}_{\geq 0}$ are known constants. Using [22, Def. 3], the generalized time-derivative of $\mathcal{V}_L$ can be computed as $\dot{\overline{\mathcal{V}}}_\sigma(z, t) \triangleq \max_{p \in \partial \mathcal{V}_L(z)} \max_{q \in F_\sigma(z,t)} p^T q$, where $\partial \mathcal{V}_L$ denotes the Clarke gradient of $\mathcal{V}_L$ defined in [23, pp. 39]. Since $z \mapsto \mathcal{V}_L(z)$ is continuously differentiable, $\partial \mathcal{V}_L(z) = \{\nabla \mathcal{V}_L(z)\}$, where ∇ denotes the standard gradient operator. Thus,

$$\dot{\overline{\mathcal{V}}}_\sigma(z, t) = \max_{q \in F_\sigma(z,t)} (\nabla \mathcal{V}_L(z))^T q \overset{a.e.}{\leq} \max_{q \in F_\sigma'(z,t)} (\nabla \mathcal{V}_L(z))^T q,$$

where the notation $\overset{a.e.}{(\cdot)}$ denotes that the relation $(\cdot)$ holds for almost all $t \in \mathbb{R}_{\geq 0}$. Additionally, using [21, Lemma E.1.IV],[4] the update law-based terms that appear after evaluating $\max_{q \in F_\sigma'(z,t)} (\nabla \mathcal{V}_L(z))^T q$ can be upper-bounded as

$$-\mathrm{vec}(\widetilde{V}_j)^T \Gamma_j^{-1} K[\mathrm{proj}](\Gamma_j \Lambda_{j,\sigma}^T e) \leq -\mathrm{vec}(\widetilde{V}_j)^T \Lambda_{j,\sigma}^T e, \tag{21}$$

$\forall j \in \{0, \ldots, k\}$. Thus, evaluating $\max_{q \in F_\sigma'(z,t)} (\nabla \mathcal{V}_L(z))^T q$, using (21) and the fact that $e^T K[\mathrm{sgn}](e) = \{\|e\|_1\}$ yields

$$\dot{\overline{\mathcal{V}}}_\sigma(z, t) \overset{a.e.}{\leq} e^T (f_e + \sum_{j=1}^k \Xi_{j,\sigma} \Delta_{j,\sigma} + \varepsilon_\sigma(x_d) - \rho(\|e\|)e - k_1 e) + \max \sum_{j=0}^k \{e^T \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j) - \mathrm{vec}(\widetilde{V}_j)^T \Lambda_{j,\sigma}^T e\} - k_s \|e\|_1. \tag{22}$$

Noting that $e^T \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j) = (e^T \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j))^T = \mathrm{vec}(\widetilde{V}_j)^T \Lambda_{j,\sigma}^T e$ since they are scalar, the term $\max \sum_{j=0}^k \{e^T \Lambda_{j,\sigma} \mathrm{vec}(\widetilde{V}_j) - \mathrm{vec}(\widetilde{V}_j)^T \Lambda_{j,\sigma}^T e\} = 0$, $\forall \sigma \in \mathcal{N}$.

[3]The assumption [22, Assumption 1] is equivalent to the assumption that ϱ is locally bounded. Since the switched system in (19) involves a finite number of subsystems, the assumption is always satisfied in this letter.

[4]The lemma says $-\tilde{\theta}^T \Gamma^{-1} \mathrm{proj}(\mu) \leq -\tilde{\theta}^T \Gamma^{-1} \mu$. This property also holds after replacing $\mathrm{proj}(\mu)$ with $K[\mathrm{proj}](\mu)$, since $K[\mathrm{proj}](\mu)$ evaluates as the set of convex combinations of $\mathrm{proj}(\mu)$ and μ, whenever $\mathrm{proj}(\mu)$ is discontinuous.

Thus, substituting $\|\sum_{j=1}^{k} \Xi_j \Delta_j\| \leq c$, $\|\varepsilon_\sigma(x_d)\| \leq \overline{\varepsilon}$, and $e^T f_e \leq \rho(\|e\|)\|e\|^2$, (22) can be upper bounded as

$$\dot{\mathcal{V}}_\sigma(z,t) \overset{a.e.}{\leq} e^T(c + \overline{\varepsilon} - k_1 e) - k_s\|e\|_1.$$

Using $-k_s\|e\|_1 \leq -k_s\|e\|$, and selecting k_s according to the theorem statement yields

$$\dot{\mathcal{V}}_\sigma(z,t) \overset{a.e.}{\leq} -k_1\|e\|^2. \tag{23}$$

By invoking [22, Th. 2], $z \in \mathcal{L}_\infty$ and $\|e(t)\| \to 0$ as $t \to \infty$. Additionally, $z \in \mathcal{L}_\infty$ implies $\widetilde{V}_j, \hat{V}_j \in \mathcal{L}_\infty$, $\forall j \in \{0, \ldots, k\}$. Since Φ is a locally essentially bounded function, it follows that $\hat{\Phi}$ is bounded. Therefore, since all terms on the right hand side of (6) are bounded, it follows that $u \in \mathcal{L}_\infty$. Moreover, since ϕ_j and ϕ_j' are locally essentially bounded functions for all $j \in \{0, \ldots, k\}$, it follows from (7) and (8) that $\dot{\hat{V}}_j \in \mathcal{L}_\infty$, $\forall j \in \{0, \ldots, k\}$. ■

IV. Simulations

Four simulation examples are provided to demonstrate the efficacy of the developed method, and the results are quantitatively compared with known baseline methods such as offline pre-training and output-layer adaptation [11]. The nonlinear system in (1) is considered with $f(x) = [x_1 x_2 \tanh(x_2) + \text{sech}^2(x_1),\ \text{sech}^2(x_1 + x_2) - \text{sech}^2(x_2)]^T$, where $x = [x_1,\ x_2]^T$. The desired trajectory is $x_d(t) = [\sin(2t),\ -\cos(t)]^T$, the initial condition is $x(0) = [1, 2]^T$, the control gains are selected as $k_1 = 20$ and $k_s = 1$, and the bound for projection operator is selected as $\overline{V} = 5000$. The DNNs in the first and second examples, i.e., DNN1 and DNN2, consist of 6 layers, with 7 neurons in each layer; hence, there is a total of 231 individual weights in the first two examples. The DNNs in the third and fourth examples, i.e., DNN3 and DNN4, consist of 10 layers, with 30 neurons in each layer; hence, there is a total of 90150 individual weights in the third and fourth examples. Each simulation is performed for 10 seconds. To prevent the DNN term from having a large initial value, the inner and output layer weights are initialized as random values from the uniform distributions $U(0, 0.5)$ and $U(0, 0.01)$, respectively.

DNN1 and DNN3 contain LReLU activation functions given by $\varsigma(y) = y$ for $y \geq 0$, and $\varsigma(y) = 0.01y$, otherwise. The adaptation gain for DNN1 and DNN3 is selected using the switched rule: $\Gamma_j = 10 I_{L_j L_{j+1}}$, if $\|[\,\text{vec}^T(\tilde{V}_0) \quad \ldots \quad \text{vec}^T(\tilde{V}_k)\,]^T\| \leq 5$, and $\Gamma_j = I_{L_j L_{j+1}}$, otherwise, for all $j \in \{0, \ldots, k\}$. DNN2 and DNN4 contain hyperbolic tangent activation functions given by $\varsigma(y) = \tanh(y)$. Unlike LReLUs, saturating activation functions like hyperbolic tangents suffer from the vanishing gradient problem [4], i.e., the gradient terms in the update law vanish as the activation function saturates, which slows down the weight updates. To compensate for vanishing gradients and for a fair comparison with the LReLU-based DNNs, the adaptation gain for the hyperbolic tangent activation function-based DNNs is selected with a relatively larger value of $\Gamma_j = 500 I_{L_j L_{j+1}}$.

Figure 1 shows the plots of DNN weight estimates, tracking error, and function estimation error for DNN3 and DNN4, where $\tilde{f} \triangleq f(x) - \hat{\Phi}$ denotes the function estimation error. The plots demonstrate that asymptotic convergence of the tracking error e is achieved in 0.5 s for both the examples. Table I provides a quantitative comparison of the developed method with offline training and output-layer adaptation [11], where e_{RMS} denotes the root mean square (RMS) of e over the time interval [0, 10], and $e_{\text{RMS,SS}}$ and $\tilde{f}_{\text{RMS,SS}}$ denote the RMS of e and $\tilde{f}$, respectively, over the time interval [5, 10] (i.e., in steady state). For the simulations in Tab. I, the robustifying term $k_s \text{sgn}(e)$ is removed to better quantitatively compare the effects of the DNN term. The offline pre-trained DNN is trained using data collected from 600 seconds of an *a priori* simulation. Using LReLUs yields improvement in the steady state tracking and function estimation performance as compared to hyperbolic tangent units as evident from the $\|e_{\text{RMS,SS}}\|$ and $\|\tilde{f}_{\text{RMS,SS}}\|$ values for DNN1 vs. DNN2 and DNN3 vs. DNN4. The developed method provides a decreased $\|e_{\text{RMS,SS}}\|$ but an increased $\|e_{\text{RMS}}\|$ as compared to offline pre-training or using adaptation for only the output-layer. This discrepancy is due

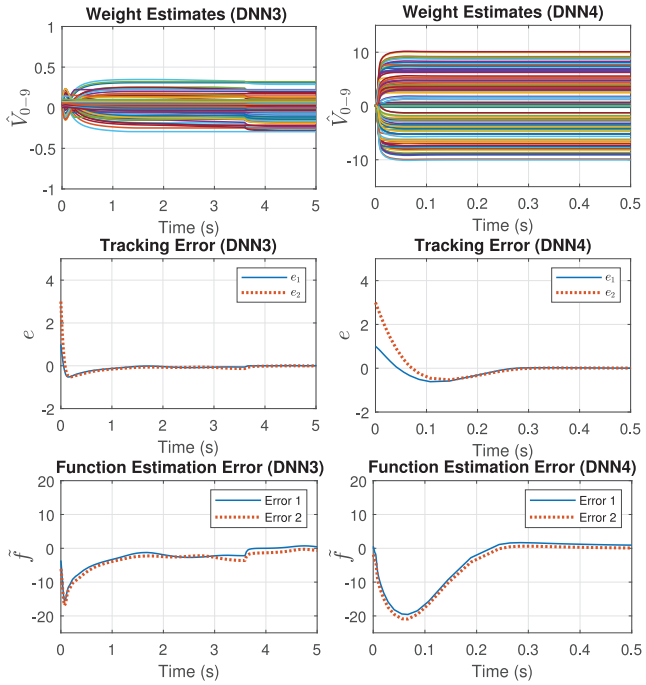

Fig. 1. Plots of DNN weight estimates, tracking error, and function approximation error for DNN3 and DNN4. The simulation is performed for 10 seconds. For a better visualization of the transient performance, the plots for LReLU and tanh are shown for 5 and 0.5 seconds, respectively. Additionally, we show 150 arbitrarily selected weight estimates out of the total 90150 weights for a tractable visualization.

TABLE I
DNN Performance Comparison

Method	$\|e_{\text{RMS}}\|$	$\|e_{\text{RMS,SS}}\|$	$\|\tilde{f}_{\text{RMS,SS}}\|$
Developed (DNN1)	0.7014	0.0059	0.1204
Developed (DNN2)	1.2687	0.0108	0.2178
Developed (DNN3)	0.4326	0.0056	0.6824
Developed (DNN4)	0.3844	0.0063	0.7823
Offline (DNN1)	0.2952	0.0216	1.2165
[11] (DNN1)	0.6831	0.0105	0.2143

to the initial overshoot in tracking error due to weight adaptation. The developed method provides a tenfold and twofold improvement in steady-state function estimation as compared to offline pre-training and output-layer adaptation, respectively.

V. Conclusion

This letter presents Lyapunov-based real-time weight update laws for each layer of a feedforward DNN. Additionally, the developed method also allows nonsmooth activation functions to be used in the DNN architecture. A nonsmooth Lyapunov-based stability analysis is provided to guarantee global asymptotic tracking error convergence. Simulation results are provided for a nonlinear system using DNNs involving leaky ReLU and hyperbolic tangent activation functions to demonstrate the efficacy of the developed method. Using LReLUs yields improvement in the steady-state tracking and function estimation performance when compared to hyperbolic tangent activation functions. Although adapting for more layers might cause initial overshoot in the tracking error, the developed method provides tenfold and twofold improvement in steady-state function estimation as compared to offline pre-training and output-layer adaptation, respectively.

Appendix

Proof of Lemma 1: We prove this lemma by mathematical induction. Using (4) and Fact 1 yields $\widetilde{\Phi}_0 = \widetilde{V}_0^T x_{da} = (I_{L_1} \otimes x_{da}^T)\text{vec}(\widetilde{V}_0)$. Since $\overset{\curvearrowleft 0}{\prod_{l=1}} \hat{V}_l^T \hat{\phi}_l' = 1$, it can be verified that (16) also yields $\widetilde{\Phi}_0 = (I_{L_1} \otimes x_{da}^T)\text{vec}(\widetilde{V}_0)$. Thus, Lemma 1 holds for $j = 0$. To use induction, assume (16) applies for $j = h-1$, given any arbitrary $h \in \{1, \ldots, k\}$, and evaluate $\widetilde{\Phi}_{h-1}$. Then, using (15) with $j = h$ yields

$$\widetilde{\Phi}_h = (I_{L_{h+1}} \otimes \hat{\phi}_h^T)\text{vec}\big(\widetilde{V}_h\big) + \hat{V}_h^T \hat{\phi}_h' \widetilde{\Phi}_{h-1} + \Delta_h \qquad (24)$$

Substituting $\widetilde{\Phi}_{h-1}$ into (24) and rearranging terms, it can be verified that the obtained expression is the same as that obtained using (16). Thus, Lemma 1 also applies for $j = h$. ■

References

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.

[4] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[5] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3866–3878, Sep. 2020.

[6] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Contr. Syst. Lett.*, vol. 2, no. 3, pp. 543–548, Jul. 2018.

[7] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3050–3057, Apr. 2020.

[8] F. L. Lewis, "Nonlinear network structures for feedback control," *Asian J. Control*, vol. 1, no. 4, pp. 205–228, 1999.

[9] G. Joshi and G. Chowdhary, "Deep model reference adaptive control," in *Proc. IEEE Conf. Decis. Control*, Nice, France, 2019, pp. 4601–4608.

[10] G. Joshi, J. Virdi, and G. Chowdhary, "Asynchronous deep model reference adaptive control," in *Proc. Conf. Robot Learn.*, 2020, pp. 984–1000.

[11] R. Sun, M. L. Greene, D. M. Le, Z. I. Bell, G. Chowdhary, and W. E. Dixon, "Lyapunov-based real-time and iterative adjustment of deep neural networks," *IEEE Contr. Syst. Lett.*, vol. 6, pp. 193–198, Jan. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9337905

[12] D. M. Le, M. L. Greene, W. A. Makumi, and W. E. Dixon, "Real-time modular deep neural network-based adaptive control of nonlinear systems," *IEEE Contr. Syst. Lett.*, vol. 6, pp. 476–481, May 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9432951

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, vol. 30, 2013, p. 3.

[14] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1319–1327.

[15] B. E. Paden and S. S. Sastry, "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators," *IEEE Trans. Circuits Syst.*, vol. 34, no. 1, pp. 73–82, Jan. 1987.

[16] D. S. Bernstein, *Matrix Mathematics*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[17] J. Cortes, "Discontinuous dynamical systems," *IEEE Control Syst. Mag.*, vol. 28, no. 3, pp. 36–73, Jun. 2008.

[18] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Proc. Conf. Learn. Theory*, 2020, pp. 2306–2327.

[19] F. L. Lewis, A. Yegildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, Mar. 1996.

[20] R. Kamalapurkar, J. A. Rosenfeld, J. Klotz, R. J. Downey, and W. E. Dixon, "Supporting lemmas for RISE-based control methods," 2014, *arXiv:1306.3432*.

[21] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic, *Nonlinear and Adaptive Control Design*. New York, NY, USA: Wiley, 1995.

[22] R. Kamalapurkar, J. A. Rosenfeld, A. Parikh, A. R. Teel, and W. E. Dixon, "Invariance-like results for nonautonomous switched systems," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 614–627, Feb. 2019.

[23] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Philadelphia, PA, USA: SIAM, 1990.